НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ Олександр Коваль
(підпис)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 122 Комп’ютерні науки та інформаційні технології

на тему Моделювання та візуалізація процесу пошкодження автомобілів

Виконав (-ла): студент (-ка) 4 курсу, групи ТР-62

Богдан Денис Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к. т. н., доц., Шаповалова С. І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Богдану Денису Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Моделювання та візуалізація процесу пошкодження автомобілів

керівник роботи ____доцент, кандидат технічних наук, Шаповалова Світлана Ігорівна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 2020р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи __візуальна мова програмування Blueprint, ігровий рушій Unreal Engine 4

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) __проаналізувати обраний ігровий двигун та програмне забезпечення для розробки, проаналізувати розроблену модель автомобіля, впровадити систему пошкодження автомобіля, впровадити систему реакції на постріли, створити візуальні ефекти

_____

_____

5. Перелік ілюстративного матеріалу

_____

_____

_____

_____

_____

6. Дата видачі завдання ”_11_ ”___жовтня___2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту		
8.	Передзахист		
9.	Захист		

Студент _____ _____Богдан Д. Ю._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Шаповалова С. І._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Записка містить 50 сторінку, 32 рисунки, 3 додатки та 5 посилань.

Мета роботи – створити систему візуалізації та пошкодження автомобілів для комп'ютерних або мобільних ігор, яка б надавала змогу, при невеликих затратах ресурсів пристрою, симулювати та візуалізувати пошкодження автомобіля при різному на нього впливу.

Вибір ігрового двигуна Unreal Engine 4 для роботи, було зроблено, оскільки даний ігровий двигун дозволяє розробляти системи пошкодження, які б виглядали реалістично, та не потребували багато ресурсів.

У результаті було розроблено систему пошкодження автомобіля з гнучким набором можливих пошкоджень та варіантів взаємодії, які можна персоналізувати під особисті потреби користувачів та розробників ігрових застосунків.

Ключові слова: ігровий двигун Unreal Engine 4, система візуалізації та пошкодження автомобілів, персоналізація системи.

ABSTRACT

The note contains 50 pages, 32 pictures, 3 appendices and 5 links.

The purpose of the work is to create a system of visualization and damage to cars for computer or mobile games, which would allow, at low cost of device resources, to simulate and visualize damage to the car with different effects on it.

The choice of the Unreal Engine 4 gaming engine to work with was made because this gaming engine allows you to develop damage systems that look realistic and do not require many resources.

As a result, a car damage system was developed with a flexible set of possible damages and interaction options that can be personalized to the personal needs of users and game developers.

Key words: Unreal Engine 4 game engine, car visualization and damage system, system personalization.

ЗМІСТ

ВСТУП……..……8
1. Постановка задачі моделювання та візуалізації..……10
2. Методи та засоби розробки…….…….……..12
2.1. Фізичні моделі……..……..…...12
2.2. Програмні засоби розробки……..……14
2.3. Мова реалізації логіки моделі…….……15
3. Програмна реалізація…….…….…...16
3.1. Визначення частин моделі автомобіля та їх реакції на постріли…..……16
3.2. Симуляція підвіски……...22
3.2.1. З’єднувачі……....22
3.2.2. Реалізація симуляції……...22
3.3. Перевірка попадань……...24
3.4. Моделювання автомобіля…….....25
3.4.1. Текстура авто……..…….…...27
3.4.2. Руйнуючі меші……..……..……....28
3.4.3. Розбиття скла…….…….....29
3.4.4. Скелетні меші……..…….....29
3.4.5. Кістки……..…….....31
3.5. VFX наслідки попадання кулі в різні частини моделі автомобіля…….…..31
3.5.1. VFX……..…….....32
3.5.2. «Декалі»……..33
3.5.3. Карти нормалі……..…….....33
3.5.4. VFX кулі на кузові автомобіля……..…...34
3.5.5. VFX злива палива……...34
3.5.6. VFX розтікання палива по поверхні…….…...34

4. Методика роботи користувача з програмною системою……35
4.1. Інсталяція системи……..…….……35
4.2. Системні вимоги…….……..35
4.3. Інструкція користувача…….……...36
4.3.1. Авто та інші блюпринт-події…….…….…….....37
4.3.2. Додавання нової моделі……..…..…….39
4.4. Сценарій роботи системи…….……...43
ВИСНОВКИ……..…..49
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……..……50
ДОДАТОК А……..…….…..…..55
ДОДАТОК Б……..……..…….……..57
ДОДАТОК В……..……..…….……..61

ВСТУП

В нинішні часи вкрай **актуально** використовувати потенціал мобільних пристроїв задля здобуття ефекту повного занурення у ігровий процесс, але при цьому розробники мобільних додатків та ігор мають значні ліміти, надиктовані характеристиками пристрою користувача. Саме тому постало питання розробки 3Д-об’єктів оточуючого нас світу та надання їм фізичних властивостей для повного чи часткового, в залежності від ресурсів пристрою, відтворення симуляцій.

Розробка такої системи дозволило б будь-якому користувачу імітувати фізичну поведінку взаємодії реальних об’єктів, при цьому не хвилюючись за ресурси системи, і з мінімальними втручанням до системи. А при потребі з легкістю налаштувати таку систему під свою потреби.

Така система повинна бути достатньо оптимізованою, задля її використання на будь-яких системах та пристроях, з мінімальним зменшенням її функціоналу або остаточного візуального вигляду.

Також така система могла б використовуватися для ознайомлення з ігровим двигуном, на якому вона розроблена. Хоч вона і не використовує всіх можливостей двигуна, але більшість основних модулів, плагінів і систем дають змогу побачити основні можливості двигуна.

Важливим елементом даної системи є те, що її концепт та основну суть можна було б відтворити на будь-якому ігровому двигуні, зі збереженням всіх її переваг. При цьому деякі з наявних плагінів присутні в багатьох ігрових двигунах, що спрощує відтворення такої системи, при потребі перенесення концепту на інший ігровий двигун.

Ця система створена як для початкових користувачів ігрового двигуна, так і для просунутих розробників ігрової індустрії. Вона може використовуватися як с ціллю повного влаштовування в ігровий проект, для симуляції взаємодії гравця та машини,

так і з ціллю часткового влаштовування в ігровий проект, с подальшою зміною 3Д моделі авто на іншу. При чому, її можна використовувати як на мобільних пристроях, так і на будь якому ПК або консолях нинішніх або подальших поколінь.

Дана дипломна робота містить чотири розділи:

- Перший розділ описує постановку задачі моделювання та візуалізації автомобіля, з поставленою метою да списком завдань, що потрібно виконати для повної реалізації системи
- Другий розділ описує методи та засоби розробки системи, що були використані для розробки цієї системи, а також фізичну модель, інструменти, плагіни, окремі модулі і мови програмування
- Третій розділ описує програмну реалізацію системи, поведінку кожного окремого модуля і процес створення системи
- Четвертий розділ описує методику роботи користувача с програмною системою, її системні вимогу, налаштування, і демонструє її роботу

Висновки містять результати досліджень, короткий перелік виконаних задач, та опис результатів виконання.

1. ПОСТАНОВКА ЗАДАЧІ МОДЕЛЮВАННЯ ТА ВІЗУАЛІЗАЦІЇ

Метою дипломної роботи є створення системи візуалізації результатів обстрілу автомобіля (Car Shooting System англ.) для використовування в комп'ютерних іграх та ігрових проектах, на будь-якій платформі, наприклад ПК, консолі або мобільні пристрої.

Для досягнення цієї мети необхідно виконати такі **завдання**:

- Розробити 3D модель машини з всіма необхідними модулями та надати їй всі необхідні кістки
- Створити фізичну модель обстрілу автомобіля, відслідковування коллізій, реагування на них
- Створити базові інтерфейси для взаємодії фізичної моделі з готовим ігровим проектом
- Провести тестування, для виявлення помилок, некоректної роботи, визначення причини їх виникнення, та їх усунення
- Створити інструкцію користувача, яка описує взаємодію користувача з системою, її налаштування і додання нової моделі

Система має бути представлена як з'єднаний між собою набір модулів, який може бути інтегрований будь-яким розробником ігрового програмного забезпечення з іншими програмними ігровими модулями та моделями, або доданий як готовий набір модулів в ігровий проект. Запропонований застосунок може використовуватися як за умов внесення незначних налаштувань моделі обстрілу автомобіля, та зміни візуальної моделі автомобіля, так і як готова система зі стандартною візуальною моделлю автомобіля.

Мають бути реалізовані такі результати обстрілу:

- зміна зовнішнього вигляду автомобіля під дією вогню

- відтворення руйнувань на деталях автомобіля
- наслідки від куль при попаданні на поверхню
- відкриття дверей та багажнику при попаданні
- відстріл дзеркал
- пошкодження корпусу фари та розбиття лампи фари з відключенням світла
- поетапний відстріл бампера від кузова
- тріскання та розбиття скла
- спускання коліс
- пробивання бензобаку, розлив пального, та згорання автомобіля

Для реалістичного відображення наслідків пострілу, повинні бути задіяні така система відображення ефектів, як VFX. Вона повинна дозволяти створити реалістичні ефекти пострілів без великого навантаження на процесор.

Систему необхідно реалізувати з використанням ігрового двигуна Unreal Engine 4, з використанням візуальної мови програмування Blueprint.

Модель автомобіля повинна бути реалізована в безкоштовному 3D редакторі Blender3D.

Для тестування та демонстрації такої системи, повинен бути створений тестовий стенд, що надавав би змогу ретельно дослідити окрему поведінку всіх окремих модулів та всієї системи в загалом.

2. МЕТОДИ ТА ЗАСОБИ РОЗРОБКИ

Система створена на двигуні Unreal Engine 4, створеному компанією Epic Games, як двигун з вільною ліцензією на використання.

Система створена виключно на візуальній псевдомові Blueprint за допомогою нод, що дозволяє легко налаштовувати та змінювати її будь-якому користувачу цієї систему при мінімальних навичках програмування.

Система розроблена на двигуні (рушій) Unreal Engine 4, розроблений компанією Epic Games. Це програмне забезпечення стало основою для формування цієї системи.

Модель автомобіля створена в 3D редакторі Blender3D, який створений компанією Blender Foundation, та оригінальним автором Тоном Розендаллем.

Введемо основні визначення термінів, які використовуються в роботі:

Нода — це такі об'єкти, як події, виклики функцій, операції управління потоком, змінні, тощо, які можуть бути використані у графі візуалізації логіки для визначення функціональності конкретного графа та блюпринта, що його містить.

Меш — це фрагмент геометрії, який складається з набору багатокутників, які можуть кешуватися у відеопам'яті та відображатись за допомогою відеокарти. Це дозволяє їх ефективно рендерувати, тобто вони можуть бути набагато складнішими, аніж інші типи геометрії, такі як браші.

2.1 Фізична модель

Фізична модель - представлення фізичної системи, явищ та процесів, з метою використання як повна або частична симуляція поведінки фізичних об'єктів.

Для створення фізичної моделі, в системі руйнування, було використано такий список плагінів, та готових систем:

- APEX Destruction Plugin
- Voronoi Destructible Plugin
- PhysX Plugin
- Cascade Particle System
- Unreal Engine 4 Physics

APEX Destruction Plugin — це плагін розроблений компанією Nvidia, та змінений компанією Epic Games, який використовується для руйнування мешів по фізичним властивостям, а також являється базовою системою для використання руйнівних мешів.

Voronoi Destructible Plugin — це плагін, розроблений компанією Epic Games, який використовується для генерування руйнівних мешів, та контролювання і налаштування цієї розбивки.

PhysX Plugin — це плагін розроблений компанією Nvidia, який використовується для багатьох обрахунків фізики, які замість процесора можуть бути прораховані на відеокарті. В цій системі він використовуєтся для невеликих фізичних об’єктів, таких як часточки з VFX та роздроблені руйнівні меші.

Cascade Particle System — це модуль, розроблений компанією Epic Games, для свого двигуна, який використовується для створення VFX, їх контролювання та відображення в редакторі і готовій грі.

Unreal Engine 4 Physics — це модуль, розроблений компанією Epic Games, для свого двигуна Unreal Engine 4, який відповідає за фізику, будь-яку фізичну взаємодію та всі фізичні об’єкти в двигуні. В системі візуалізації пошкодження автомобілів він відповідає за з'єднувачі, реакцію на коллізію та скелетні меші.

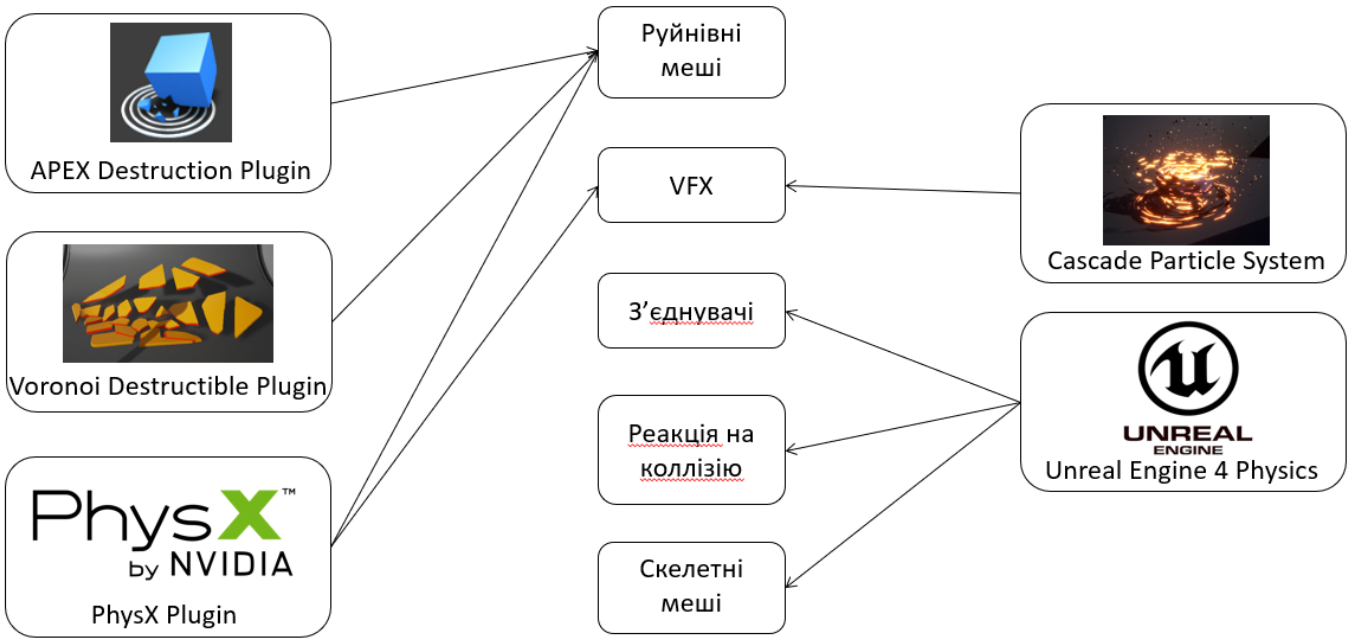


Рисунок 2.1 — діаграма взаємодії модулів і плагінів з об'єктами системи

На рисунку 2.1 продемонстровано діаграму взаємодії окремих модулів і плагінів двигуна, з об'єктами системи візуалізації пошкодження автомобілів.

2.2 Програмні засоби розробки

Для моделювання мешу автомобіля, було використано програмний застосунок Blender3D**,** як один із найсучасніших та актуальних 3D редакторів для створення ігрових мешів.

Blender3D — це безкоштовний та відкритий набір для створення 3D-файлів. Він підтримує весь 3D-конвеєр - моделювання, такелаж, анімація, візуалізація, композиція, скульптинг та відстеження руху, редагування відео та 2D-конвеєр анімації. Перевага цієї програми полягає в тому, що її можна використовувати для навчання студентів, також вона має безліч безкоштовних плагінів, і вона постійно оновлюється та підтримується вільними розробниками.

Для реалізації самої фізичної моделі було використано актуальний на сьогоднішній день ігровий рушій Unreal Engine 4, так як він містить достатню кількість готових інструментів та плагінів для реалізації завдання.

Unreal Engine 4 (UE4) — ігровий двигун, який розробляється, підтримується і оновлюється компанією Epic Games. Він чудово підходить як для новачків ігрової розробки, так як містить легку візуальну мову програмування Blueprint, так і для досвідчених розробників, так як може програмуватися на мові програмування C++.

2.3 Мова реалізації логіки моделі

Система створена за допомогою візуальних нод, візуальною мовою програмування – Blueprint, як просту та зручну мову, особливо для налаштування системи різними користувачами. Також вона зручна для швидкого концептування.

Мета Blueprint - дати розробникам можливість оптимізувати процес налаштування інфраструктури, щоб вони могли витрачати більше часу на втілення своїх ідей і проектів.

Суттєвою перевагою Blueprint є те, що вона може використовуватися будь-яким розробником, навіть тим, хто розуміє базову логіку програмування, але не має достатньо знань та навичок мов програмування. Тобто розробник зі знаннями та навичками роботи з Blueprint зможе підкоригувати поведінку системи під свої потреби та цілі.

Blueprint — це повна система сценаріїв геймплея, заснована на концепції використання інтерфейсу на основі вузла для створення елементів ігрового процесу з Unreal Editor. Як і у багатьох поширених мовах сценаріїв, він використовується для визначення об'єктно-орієнтованих класів або об'єктів в двигуні. Використовуючи UE4, об'єкти, визначені як Blueprint, розмовно називаються лише "блюпринт".

Плагіни для Unreal Engine 4 були створені сторонніми компаніями, наприклад Nvidia, на швидкій компілюємій мові програмування C++.

C++ — мова програмування, створена як розширена версія C, яка використовується на багатьох пристроях та обчислювальних машинах.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Визначення частин моделі автомобіля та їх реакції на постріли

Формалізований опис реакції на постріли наведено в таблиці 3.1.

Таблиця 3.1. Реакції при попаданні куль до відповідних частин конструкції автомобіля

Частина конструкції автомобіля	**Реакція на постріл**	**Засоби інтерпретації**
Кузов	Сліди від куль при попаданні	Якщо в системі зареєстровано подію “Hit” для машини, від цієї події берутся координати, де було зареєстровано цю подію, та вектор нормалі по відношенні до машини в цій точці. Далі за отриманими координатам створюється VFX, вектор вершини якої буде вектором нормалі до кузова
Двері	Відкриття при першому попаданні та відхилення вліво та вправо при подальших попаданнях	Якщо в системі зареєстровано подію “Hit” для області колізії дверей, то з компонентів машини видаляється компонент з'єднувача дверей з машиною, який перебував в повністю

Продовження таблиці 3.1

		заблокованому стані до моменту впливу. Далі береться вектор попадання, і по його напрям-вектору приміняється імпульс з не великою силою до дверей
Багажник	Відкриття при першому попаданні та відхилення вниз та вгору при подальших попаданнях	Якщо в системі зареєстровано подію “Hit” для області колізії багажника, то із компонентів машини видаляється компонент з'єднувача багажника з машиною, який перебував в повністю заблокованому стані. Далі, береться вектор попадання, і по його напрям-вектору приміняється імпульс з невеликою силою до багажника
Бокові дзеркала заднього виду	Відстріл дзеркал при першому попаданні	Якщо в системі зареєстровано подію “Hit” для дзеркала, то об’єкт дзеркала віддіяється від машини, і на ньому активуються властивості симуляції фізики та колізії з машиної
Внутрішнє дзеркало заднього виду	Відстріл дзеркала першому попаданні	Якщо в системі зареєстровано подію “Hit” для дзеркала, то об’єкт дзеркала віддіяється від машини, і на ньому активуються властивості симуляції фізики та

Продовження таблиці 3.1

		колізії з машиної
Фари	Пошкодження корпусу фари при першому попаданні та розбиття лампи фари з відключенням світла при наступному попаданні	Якщо в системі зареєстровано подію “Hit” для фар, від цієї події берутся координати, де було зареєстровано цю подію, та вектор нормалі по відношенні до фари в цій точці. Далі по цим координатам створюється VFX, вектор вершини якої буде вектором нормалі до події. Потім замість статичного мешу фари з'являється руйнуючий меш (destructible mesh), до якого застосовуються пошкодження (apply damage), вектор якого співпадає з вектором попадання в фару. В кінцевому стані видимість джерела світла у фарі вмикається.
Передній та задній бампери	Відстріл бампера від кузова з фізичними коливаннями при двох попаданнях - з правої та з лівої сторони	Якщо в системі зареєстровано подію “Hit” для однієї з областей колізії бампера, то з’єднувач до якої прив'язана область видаляється із компонентів, а з’єднувач по іншу сторону бампера розблоковується, а його

Продовження таблиці 3.1

		стан з'єднання встановлюється як “Вільне”. В кінці, объект машини та бамперу поступає команда “Прокинутись”, для того щоб вони фізично відреагували на постріл
Скло (лобове, бокові передні, бокові задні, заднє)	Тріскання скла при першому попаданні, та розбиття скла на уламки при наступному	Якщо в системі зареєстровано подію “Hit” для скла, то при першому пострілу матеріал мешу, по якому було здійснено постріл змінюється на потрісканий. При наступному попаданні, від цієї події берутся координати, де було зареєстровано цю подію, та вектор нормалі по відношенні до скла. На місці статичного меша скла, створюється руйнуючий меш (destructible mesh), який, в властивостях якого включається симулювання фізики, та до якого приміняється пошкодження з вектором попадання та невеликим імпульсом. Також за цими координатами створюється VFX зі невеликими осколками скла, вектор вершини якої буде

Продовження таблиці 3.1

		вектором нормалі до скла
Шина	Спускання колеса при попаданні в шину	Якщо в системі зареєстровано подію “Hit” для шини, від цієї події берутся координати, де було зареєстровано цю подію, та вектор нормалі по відношенні до шини в цій точці. Далі за отриманими координатам створюється VFX з тиском повітря, вектор вершини якої буде вектором нормалі до шини. Також меш шини в яку було здійснено постріл, змінюється на меш пробитої шини
Диск колеса	Сліди від куль при попаданні	Якщо в системі зареєстровано подію “Hit” для диска колеса, від цієї події берутся координати, де було зареєстровано цю подію, та вектор нормалі по відношенні до диску в цій точці. Далі за отриманими координатам створюється VFX з металевими іскрами та пошкодженням металу від кулі, вектор вершини якої буде вектором нормалі до диску

Продовження таблиці 3.1

Бензобак	Пробивання бензобаку при першому попаданні та розлив пального, підпал пального та машини при наступному попаданні, згоряння машини до стану повного обгорання кузова автомобіля, вибух кожною шини по черзі, тим самим, симулюючи вибух від надмірного тиску, викликаного різким підвищенням температури кузова автомобіля та появою значною різницею внутрішнього тиску шини та зовнішнього навколишнього середовища, перебої в електропроводці під час її згоряння,	Якщо в системі зареєстровано подію “Hit” для області колізії бензобаку, то в центральній точці області колізії з вектором направленим під невеликим кутом вниз створюється VFX з потоком пального, який утворює калюжу на землі. Якщо в системі зареєстровано подію “Hit” для області колізії калюжі с пальним, то активується VFX з вогнем в області калюжу, а через секунду після, створюються такі ж самі VFX (11 штук) навколо всієї машини. Далі через деяке время всі меші скла на машині змінюють свій матеріал з цілого на потрісканий. А також протягом деякого часу, матеріал мешу кузова машини змінюється і покривається іржею, завдяки параметру “Rust” в матеріалі.

Продовження таблиці 3.1

	та миготіння фар через перебої	

Таким чином було визначено всі частини автомобіля, їх реакції на постріли та програмна реалізація кожної окремої частини.

3.2 Симуляція підвіски

3.2.1 З’єднувачі

З`єднувачі (Constraint Component) - фізичні компоненти, що використовуються задля обмеження фізичної взаємодії та пересування одного об’єкта відносно іншого.

Ці компоненти з’єднання та обмеження діють так само, як і актори (actors) для з’єднання або обмеження на сцені, за винятком того, що вони використовуються у Blueprint або можуть бути створені в C++. Враховуючи гнучкість Blueprint та потужність C++, можливо з’єднувати або обмежувати практично будь-яке фізичне тіло.

Завдяки цим компонентам, можливо симулювати реальні фізичні з’єднання, наприклад: дверні завіси, пружини, амортизатори та кріплення фізичних тіл, які створені з деформуючих матеріалів, як це представлено в цій системі.

3.2.2 Реалізація симуляції

Підвіска автомобіля створена з використанням чотирьох з'єднувачів. Вони сполучають колеса з кузовом автомобіля та налаштовані таким чином, щоб здвигати об'єкти між собою тільки по вертикальній осі. Також ці з'єднувачі налаштовані так, щоб імітувати еффект пружинної амортизації.

На рисунку 3.1 продемонстровані налаштування одного зі з'єднувачів цієї підвіски.

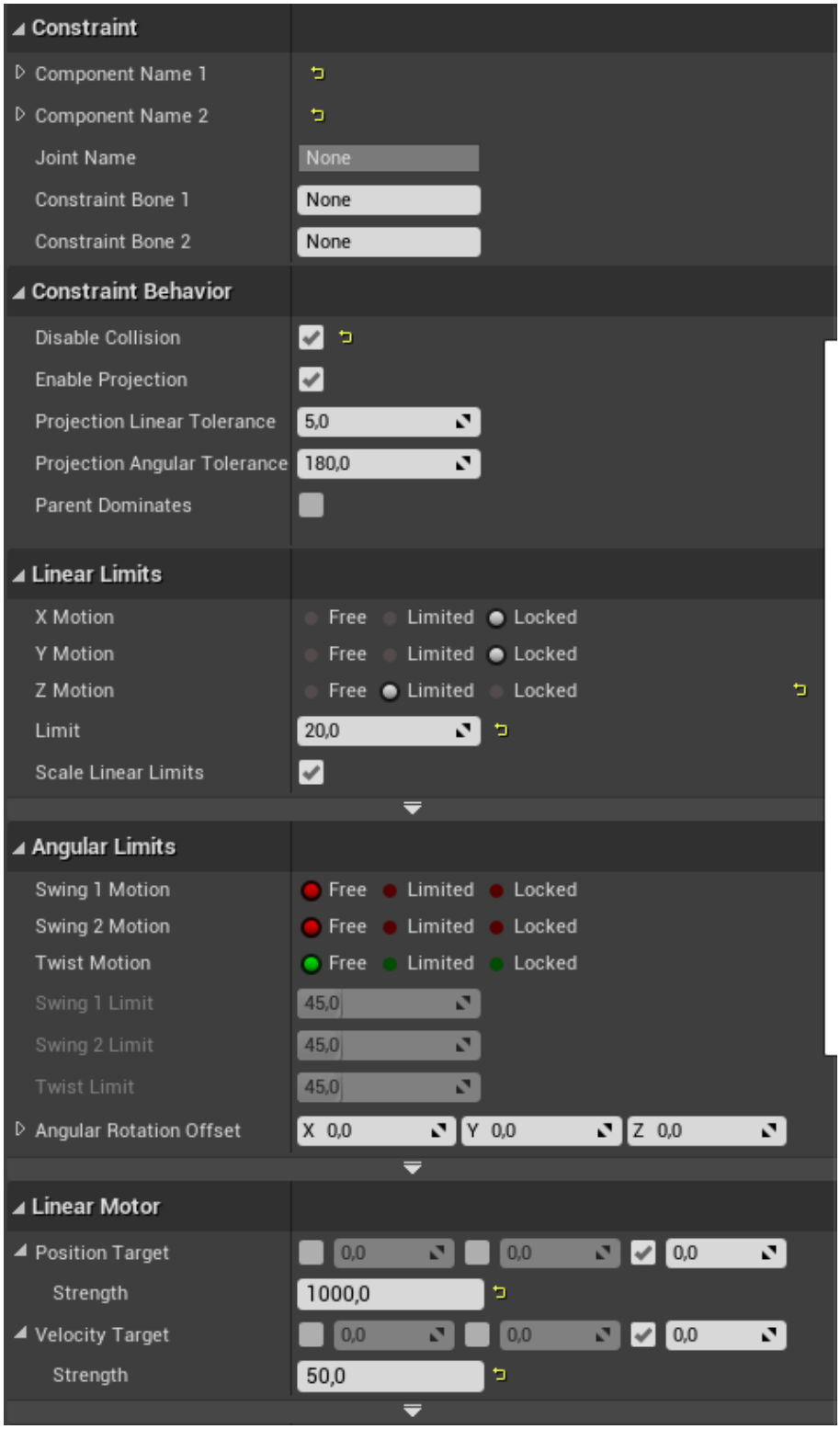


Рисунок 3.1 — Налаштування з’єднувача, який імітує підвіску автомобіля

Таким чином, можна кожну окрему пружину автомобіля можна налаштувати для задання жорсткості або м’якості підвіски.

3.3 Перевірка попадань

Для того щоб провести перевірку, чи попала куля в той чи інший елемент конструкції автомобілю, в Unreal Engine 4 використовуються коллайдери колізії. Коллайдер колізії створюється автоматично під час імпортування мешу, або створений вручну колайдер в 3D-редакторі і імпортований в двигун, задля перевірки попадання по мешу зі складною геометрією. Тобто для простої геометрії: сфера, круг, паралелепіпед, куб і тому подібних можна використовувати автоматичну генерацію колайдера колізій, а для високо деталізованих моделей краще використовувати змодельований вручну коллайдер колізії.

На рисунку 3.2 зображено автоматично згенерований коллайдер колізії меша шини, а на рисунку 3.3 зображено створений в ручну коллайдер колізії меша переднього бампера, розділений на 2 частини, для перевірки попадання кулі в окремі зони бамперу.

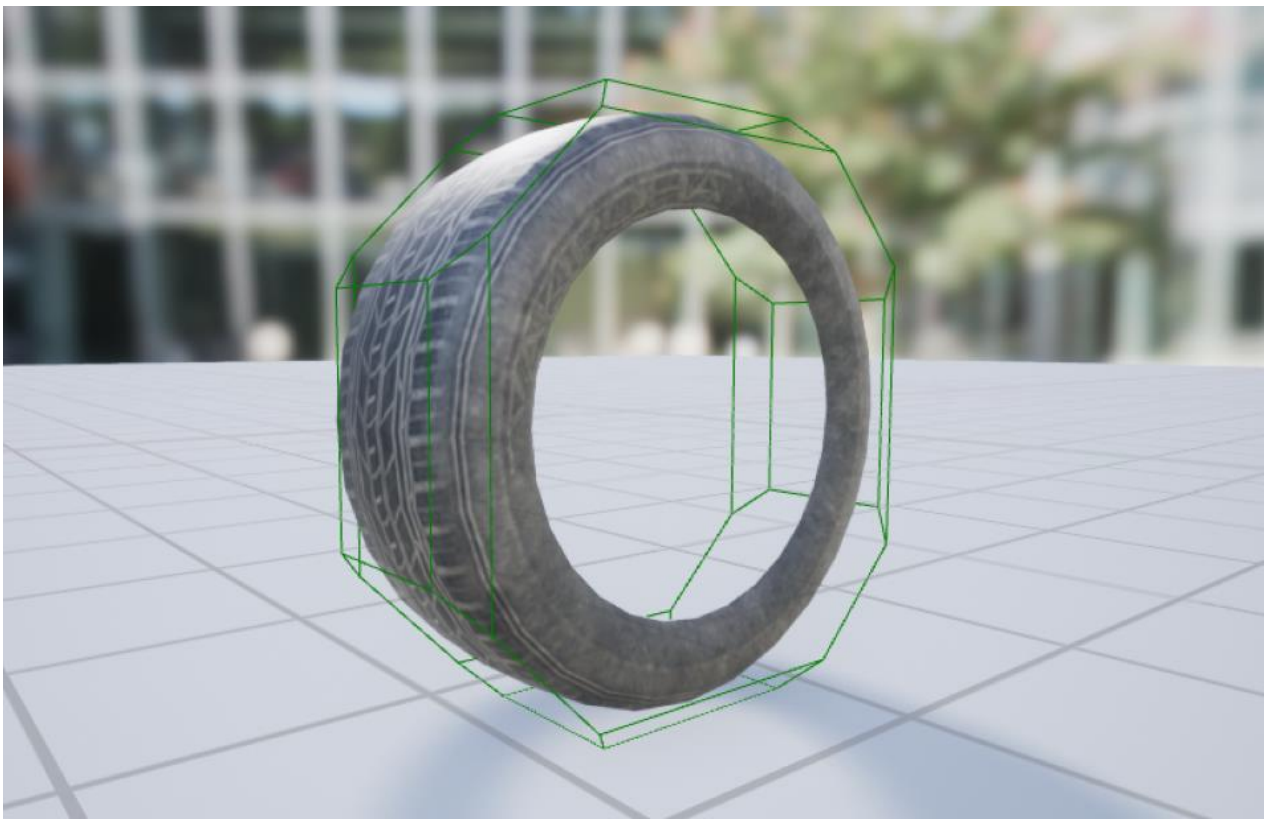


Рисунок 3.2 — Автоматично сгенерований коллайдер колізії меша шини

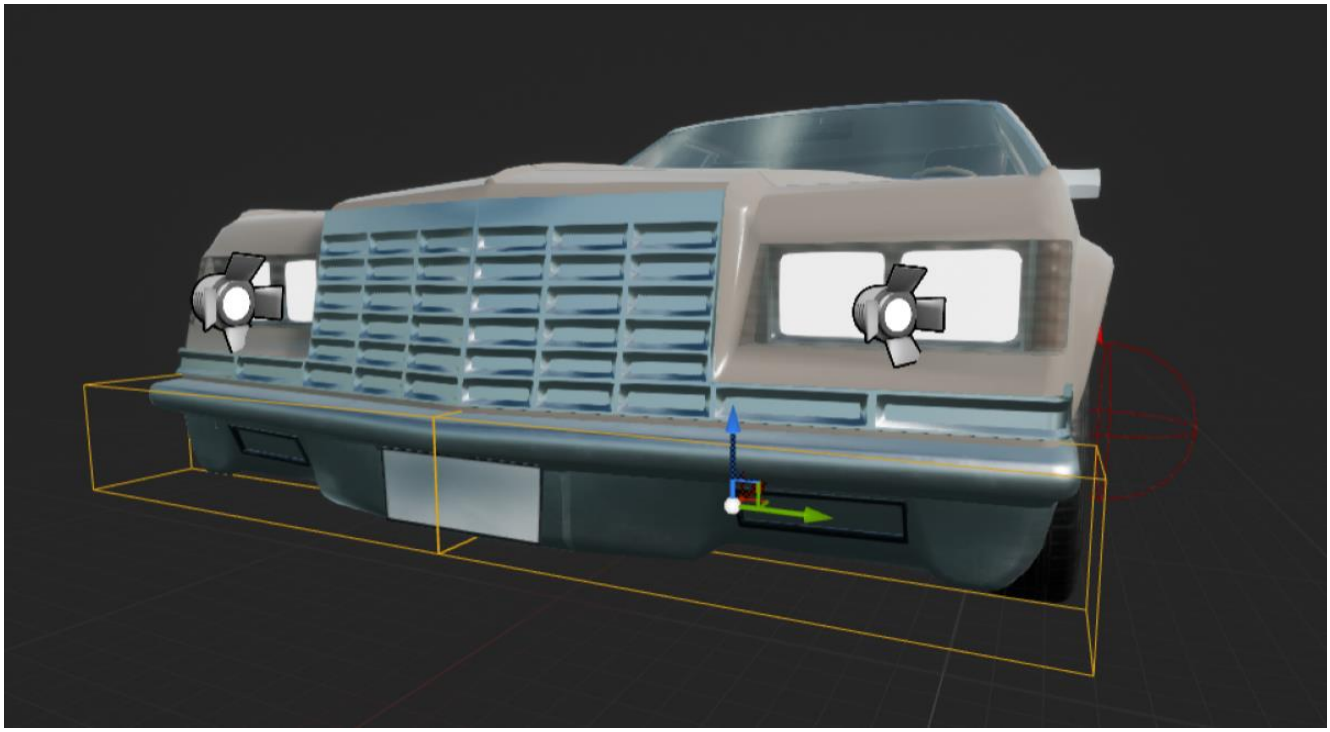


Рисунок 3.3 — Створений в ручну коллайдер колізії меша переднього бампера

Ці колайдери зазвичай складаються з примітивних фігур, таких як паралелепіпеди та циліндри.

3.4 Моделювання автомобіля

Після моделювання всіх елементів конструкції моделі автомобіля, було створено такі елементи, як окремі модулі основної системи:

- кузов
- двері
- багажник
- бокові дзеркала заднього виду
- внутрішнє дзеркало заднього виду
- передні та задні фари
- передній та задній бампер
- скло

– колеса
– шини

Фари сконструйовані з трьох частин:

– лампа фари, яка має кріплення до кузова автомобіля
– пластини з емісійним матеріалом, який випускає потік світла
– скло самої фари

Для візуалізації ефекту попадання кулі у шину було розроблено 2 моделі шин:

– звичайна підкачана шина
– здута шина

На рисунку 3.4 і 3.5 відповідно, зображено меші звичайної підкачаної шини та здутої шини.

Рисунок 3.4 — Меш звичайної підкачаної шини

Рисунок 3.5 — Меш здутої шини

Перша модель заміняється другою при взаємодії з об’єктом нанесення урону, яким може виступати куля чи вогонь. Це призводить до того, що підвіска автомобіля просідає на одну із сторін.

3.4.1 Текстура авто

На автомобіль накладені примітивні матеріали, а також поверх основної текстури було накладено текстуру з постефектом горіння - текстура обвуглення.

На початку роботи програмного забезпечення, текстура обвуглення має налаштування прозорості (alpha) зі значенням 0%. А після початку процесу горіння, значення цього параметру збільшується з часом і доходить до відмітки в 100%.

На рисунку 3.6 зображено майстер-текстуру автомобіля, яка має скалярний параметр обвуглення.

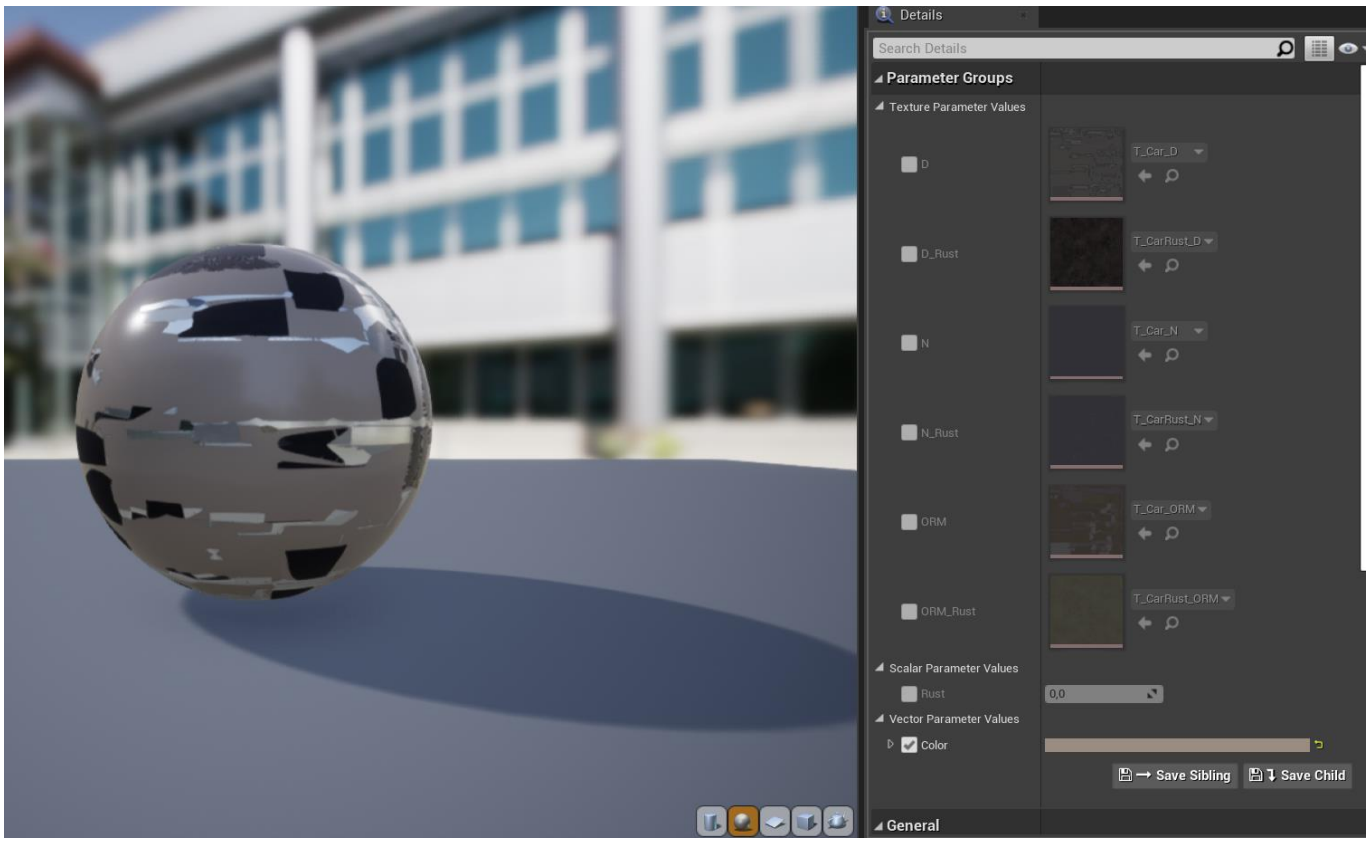


Рисунок 3.6 — Майстер-текстуру автомобіля

Завдяки цим параметрам можна змінювати матеріал текстури процедурно, з логіки системи, або вручну.

3.4.2 Руйнуючі меші

Руйнуючі меші - це меші, які мають автоматично згенеровану карту руйнації. Карта руйнації генерується за допомогою спеціального ключа, який вказує користувач, також користувач вказує кількість частинок, на яку поділиться об’єкт після зіткнення. Кожна з частинок має свій унікальний набір колайдерів колізії з налаштуванням підтримки елементів та реакцію на взаємодію з зовнішніми об’єктами.

На рисунку 3.7 зображено руйнуючий меш передньої лівої фари.

Рисунок 3.7 — Руйнуючий меш передньої лівої фари

Руйнуючим мешем може стати будь-який статичний меш, який імпортовано за замовчуванням.

3.4.3 Розбиття скла

Меш скла є статичним за замовчуванням та має стандартну текстуру скла з налаштуваннями прозорості (alpha). При першому попаданні кулі меш скла залишається статичним, але матеріал текстури змінюється на інший - з дрібними тріщинками. При другому попаданні - меш переходить у стан руйнівний меш, його текстура з попереднього кроку не змінюється, але з’являються фізичні уламки скла.

3.4.4 Скелетні меші

Скелетні меші - це меші які мають полігональну сітку та скелет, який в свою чергу побудований з кісток, завдяки яким вони можуть змінювати розташування окремих своїх елементів, прив’язаних до головної кістки. Якщо взяти за аналогію приклад з анатомії людини, то головною кісткою може бути хребет (центр маси тіла),

а дочірніми до нього будуть таз, ключиця, череп, ребра і так далі. Також скелетні меші можуть мати дочірні кіски другого, третього і так далі порядку. За аналогією: головна кістка - хребет, дочірня до неї - таз, дочірня до тазу - стегнова кістка.

Скелетні сітки складаються з двох частин:

- набір многокутників, складених для побудови поверхні скелетної сітки
- ієрархічний набір взаємопов'язаних кісток, які можуть бути використані для анімації вершин багатокутників. Зв'язуючим елементом між кістками є суглоби (на англійській joints), які дозволяють змінювати кут нахилу однієї кістки відносно іншої, що дозволяє легко маніпулювати цими елементами моделі при підготовці до анімації об'єкту (rigging)

Скелетні сітки часто використовуються в Unreal Engine 4 для представлення символів або інших анімаційних об'єктів. 3D-моделі, підготовка до анімації та анімація створюються у зовнішньому додатку для моделювання та анімації (3DSMax, Maya, Softimage, Blender3D, Houdini, тощо), а потім імпортуються в двигун Unreal Engine 4 та зберігаються в пакетах за допомогою браузера вмісту Unreal Editor.

На рисунку 3.8 зображений скелетний меш стандартної машини представленої системи.

Рисунок 3.8 — скелетний меш автомобіля

Завдяки таким скелетам, можна створювати завчасно записані анімації та фізичні анімації, які змінюються під дією різних фізичних сил.

3.4.5 Кістки

Меші скла, бокових дзеркал заднього виду були закріплені за кісткою мешу дверей відповідно: передні двері разом з дзеркалом та склом відкриваються, а задні двері разом зі склом.

Головним параметром приєднаної до мешу кістки є вага. В системі за замовчуванням встановлене значення 1 параметру ваги. Це означає, що меш буде повністю повторювати рух кістки. Якщо кут відхилення кістки становитиме, наприклад, 30°, то і кут відхилення мешу також становитиме 30°. Це видно при відкритті дверей чи багажника. Якщо встановити цей параметр на значення 0.5, то кут відхилення мешу становитиме половину кута відхилення кістки. Тобто при куті відхилення кістки у 90°, меш повернеться на 45°. Зі зниженням значення цього параметру, прямо пропорційно збільшується “вага” об’єкту маніпуляції. Важливо зазначити, що розробник та користувач системи проводить будь-які маніпуляції з автомобілем завдяки не самого мешу, а кістки, яка приєднана до мешу.

Якщо в системі не використовувати скелет, то це призведе до нестикування елементів, які можна відчинити, тому що кожна така деталь буде імпортована окремо і її потрібно буде кожен раз встановлювати в кузов автомобіля вручну.

3.5 VFX наслідки попадання кулі в різні частини моделі автомобіля

В системі такі VFX позначені ім’ям “impact” та визначено 3 таких ефекти впливу:

- розліт маленьких частинок скла
- отвір від попадання кулі в обшивку салону
- іскри та слід від попадання кулі в обшивку кузова

Проаналізувавши більшість систем в фільмах та масштабних комп'ютерних іграх, можна звернути увагу, що постріл по бензобаку викликає моментальний вибух автомобіля, однак у реальному житті це реалізувати вкрай складно, не використовуючи потужної вогнепальної зброї, наприклад, ручних пускових установок. Натомість, в представленій системі, постріл по бензобаку передбачає лише витік пального. А вже при повторному пострілі в пальне, воно запалюється. Після чого полум'я розповсюджується по всьому автомобілю, що є більш реальною поведінкою при пострілі по бензобаку автомобіля.

Саме тому було створено два VFX для бензобаку:

- злив палива
- розтікання палива по поверхні

3.5.1 VFX

VFX - система візуальних частинок, що дозволяє створювати візуальні ефекти при невеликих затратах процесорного часу, так як більшість частинок регулюється спрощеною фізичною моделлю, а відображенням цих частинок майже повністю займається відеокарта.

Unreal Engine містить надзвичайно потужну та надійну систему частинок, що дозволяє художникам створювати вражаючі візуальні ефекти, починаючи від диму, іскри та вогню до набагато більш масштабних прикладів.

Системи частинок Unreal редагуються через Cascade, повністю інтегрований та модульний редактор ефектів частинок. Cascade пропонує зворотний зв'язок у режимі реального часу та редагування модульних ефектів, що дозволяє швидко та легко створювати навіть найскладніші ефекти.

VFX також дуже тісно пов'язані з різними матеріалами та текстурами, що застосовуються до кожної частинки. Основна робота самої системи частинок - контролювати поведінку частинок, тоді як вигляд системи в цілому часто контролюється за допомогою матеріалів.

На рисунку 3.9 відображений редактор візуальних частинок Cascade, з візуальними частинками розлива пального.

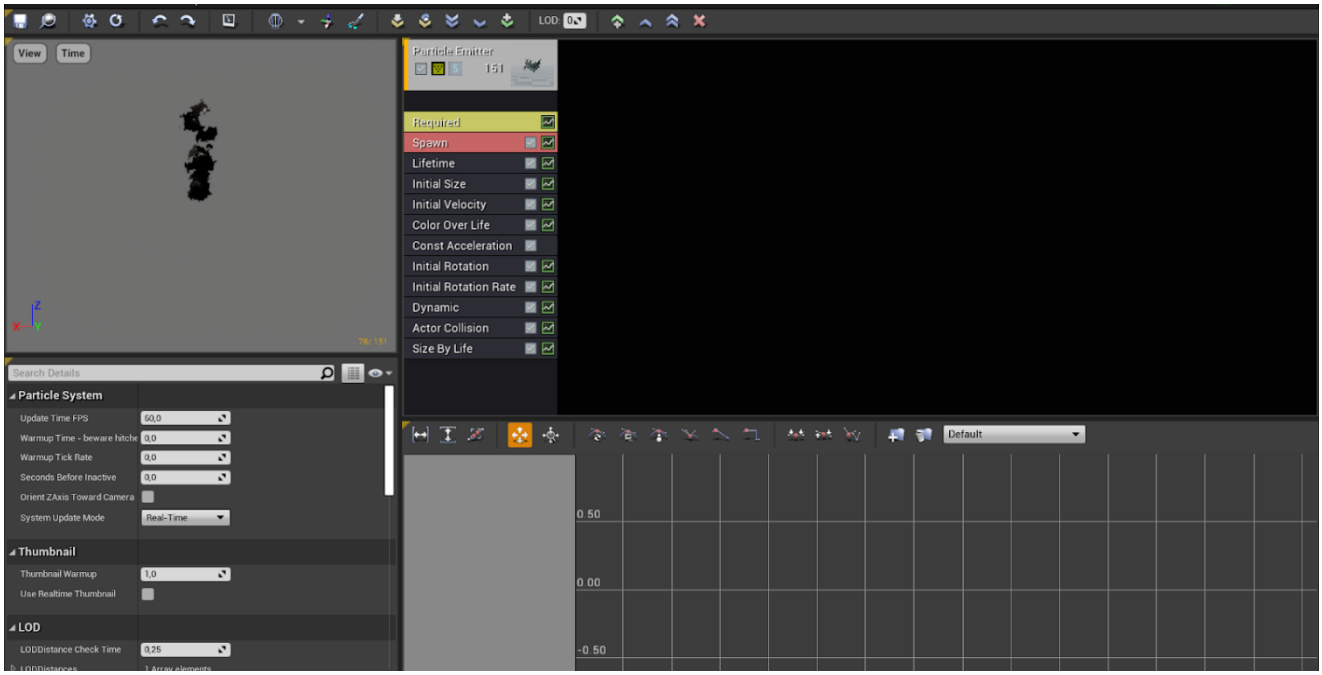


Рисунок 3.9 — Редактор візуальних частинок Cascade

Завдяки такому редактору можливо створювати не складні VFX навіть не будучи спеціалістом зі спецефектів.

3.5.2 «Декалі»

Декалі — це матеріали, які проектуються на сітки вашого рівня, включаючи статичні та скелетні сітки. Ці сітки можуть мати налаштування мобільності статичного або рухомого, і декал все ще буде проектуватися на них. Багато декалів можна відразу подавати без значного зниження продуктивності. Продуктивність зменшується із збільшенням розміру екрану та більшою кількістю інструкцій шейдера.

3.5.3 Карти нормалі

Карта нормалі — це тип карти рельєфу. Вони являють собою особливий вид текстури, що дозволяє додати деталі поверхні, такі як: удари, виїмки та подряпини на модель, за рахунок розсіювання світла по текстурі так, ніби вони представлені реальною геометрією.

3.5.4 VFX кулі на кузові автомобіля

Для спрощення симуляції попадання кулі в корпусу автомобіля, замість деформації мешу автомобіля, що є багатовитратною задачею для ігрового двигуна, були використані декалі. Декалі відкидають потрібність в заміні, або деформації мешу, так як візуально, завдяки картам нормалі, отвір від кулі здається реальним та об’ємним, через імітування розсіювання світла. Також декалі дозволяють оминути обрахунки кутів вигину корпусу автомобіля, якщо куля летить під кутом до поверхні зіткнення за рахунок того, що декалі наносяться на поверхню як “наліпка”, огинаючи всі нерівності поверхні.

3.5.5 VFX злива палива

Використовуються частинки з заданою текстурою палива, які при витіканні обертаються навколо своєї осі, створюючи 3D-ефект, хоча самі частинки є 2D пластинками. При зіткненні частинки і поверхні, вона зникає, тим самим економить ресурси обчислювальної машини та “переходить” у наступний VFX.

3.5.6 VFX розтікання палива по поверхні

Текстура пального масштабується на протязі деякого часу. А після при попаданні кулі в встановлений тригер бокс (trigger box) відбувається підпал палива, що викликає третій VFX стандартного UE4 вогню.

4. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

4.1 Інсталяція системи

Для встановлення цієї системи в ігровий двигун потрібно:

1. Завантажити ігровий двигун Unreal Engine 4 безкоштовно на офіційному сайті або завантажити початковий код на GitHub з проектом та скомпілювати двигун самостійно
2. В лаунчері Epic Games зайти на вкладку Marketplace, знайти в пошуку систему Car Shooting System і підписатися на неї
3. В лаунчері Epic Games зайти на вкладку Projects, потім внизу знайти цю системи, натиснути кнопку Add to project, і вибрати свій проект, куди потрібно додати систему
4. Запустити свій проект і знайти клас машини в пошуку ассетів внизу двигуна
5. Пересунути клас машини на ігрову сцену, та розмістити її в потрібному місці
6. Налаштувати систему під свої потреби, корегуючи їх в властивостях об’єкта машини, що знаходиться на сцені

4.2 Системні вимоги

Unreal Engine 4 має наступні вимоги:

- Настільний ПК або Mac
- Windows 7 64-розрядний чи Mac OS X 10.9.2 або новішої версії
- Чотириядерний процесор Intel або AMD, 2,5 ГГц або швидше

- Карти серії NVIDIA GeForce 470 GTX або AMD Radeon 6870 HD або новішої версії
- 8 ГБ оперативної пам’яті
- 15 ГБ пам’яті на жорсткому диску, або більше
- Комп’ютерна миша та клавіатура

4.3 Інструкція користувача

Система має мати готовий клас інтерфейсу для стрільби. Це дозволить легко інтегрувати застосунок, використовувати її з будь-якими снарядами або іншими об’єктами, клас яких буде містити цей інтерфейс та почати ним користуватися.

Ядром управління системою обстрілу автомобіля є блюпринт Car_BP, об’єкт актора, який включає в себе модель автомобілю з логікою стрільби та параметрами контролю системи обстрілу.

Корисні функції:

- Користувач програмного застосунку має можливість нищити автомобіль Car_BP, будь-якими об’єктами, заздалегідь запровадивши вже готовий інтерфейс стрільби (Shooting interface) в класс кулі або вогнепальної зброї та викликавши подію “Hit from Shot event”.
- Проста та інтуїтивно зрозуміла адаптація системи під авто. Система потребує внесення трьох основних компонентів:
  - скелету
  - руйнівних мешів
  - статичних мешів

  Згрупувавши всі деталі автомобіля у відповідні категорії в блюпринті Car_BP та називаючи їх таким же самим чином, як стандартні компоненти, які встановлені за замовчуванням в програмному застосунку, зміна моделі авто не є важким завданням. Змінювати та вносити правки можна

у вже існуючі дочірні компоненти: CabinItem_BP, Carlight_BP, Mirror_BP, Wheel_BP, Windshield_BP.

4.2.1 Авто та інші блюпринт-події

В програмному застосунку необов'язково використовувати будь-які з цих подій. Будь-яка взаємодія з моделью автомобіля відбувається при зіткненні (колізії) кулі в яку імплементовано интерфейс “Shooting interface”, або з якої викликано подію “Hit” та авто. Після внесення змін потрібно перетягнути Car_BP на сцену для активації роботи події.

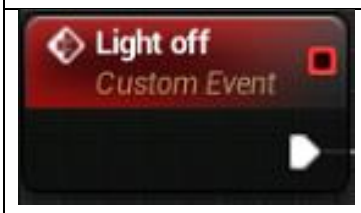
Єдина подія, котру користувачу потрібно викликати, це подія “Hit from Shot” в блюпринті Car_BP та інших системних блюпринтах.

Користувачу потрібно викликати цю подію ззовні (наприклад, в логіці класса кулі) для взаємодії з частинами конструкції автомобіля.

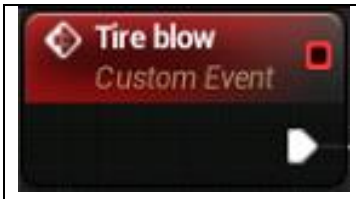
Користувачу потрібно викликати цю подію з класа кулі/пістолета (або з будь-чого іншого користувацького “зброї”), перевіривши реалізацію інтерфейсу удару (якщо об’єкт впливу не є стандартним автомобілем системи) та взаємодії з автомобілем.

Користувачу не потрібно викликати будь-які нижче наведені події (якщо вони не використовуються для зміни зовнішнього ефекту), оскільки ці події використовуються лише всередині системи.

Таблиця 4.1 — події, їх опис та графічне зображення

Графічне зображення ноди події	**Опис події**
	Подія “On Wheel Hit” викликається з блюпринта Wheel_BP, коли відбувається попадання кулі в колесо. Після зміни мешу шини, система оновлює підвіску коліс. Цю подію потрібно викликати лише у випадку зміни мешу шини
	Подія “Light off” в блюринті Carlight_BP викликається в Car_BP задля того, щоб увімкнути чи вимкнути джерело світла. Не має потреби викликати цю подію, лише тільки якщо користувачу програмного застосунку не потрібно перемкнути стан світла на постійно увімкнений або на постійно вимкнений

Продовження таблиці 4.1

	Подія “Tire blow” в блюпринті Wheel_BP викликається в Car_BP задля дефляції (спуску повітря з шини) при горінні автомобіля. Не потрібно викликати цю подію, лише тільки якщо користувачу програмного застосунку не потрібно спустити повітря з шин вручну, наприклад якщо сам гравець взаємодіє с колесом автомобіля (спускає або накачую)

4.2.2 Додавання нової моделі

Можлива зміна моделі автомобіля при деяких налаштуваннях, для використання системи з іншою машиною. Для цього потрібно надати системі всі відповідні статичні та динамічні меші (які мають здатність руйнуватися під дією зовнішніх сил).

Меші групуються у відповідні категорії в головному блюпринті, а саме:

- кузов
- колеса
- бампери
- фари
- дзеркала
- сидіння

– приборну панель
– кермо
– лобове скло

Для деталей салону автомобіля, фар, коліс, дзеркал, лобового скла потрібно розмістити статичні та руйнуючі меші в компонентах дочірніх елементів із конкретними назвами в категорії «Default» Child Actor, що знаходиться в Car_BP, рисунок 4.1

Рисунок 4.1 — Список компонентів Child Actor, що знаходиться в Car_BP

Якщо користувачеві не потрібна логіка відхилення бампера для автомобіля або у нього немає бамперів, можливо видалити логіку "Detaching bumpers on hit" в Car_BP, рисунок 4.2

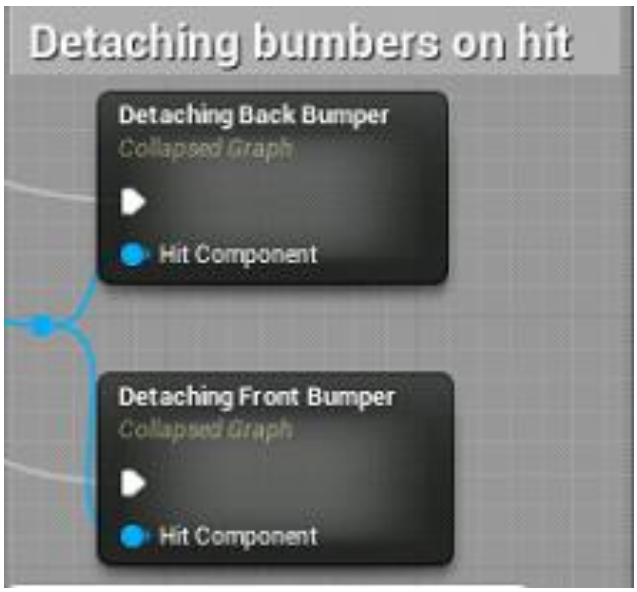


Рисунок 4.2 — Логіка "Detaching bumpers on hit" в Car_BP

Щоб забезпечити тіло скелетної сітки дверима та кістками багажника, потрібно назвати кістки так само, як у Car_Skeleton, рисунок 4.3.

Якщо у машини інша кількість дверей (у моделі, представленій в цій роботі четверо дверей), потрібно внести додаткові кістки та корективи в Car_BP, видаливши логіку, яка не використовується.

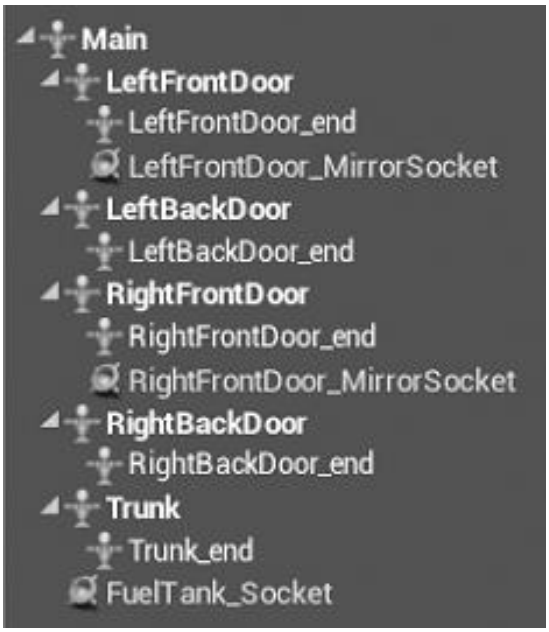


Рисунок 4.3 — Список кісток в Car_Skeleton

Потім потрібно змінити лінковані компоненти в з’єднувачах в блюпринті Car_BP, рисунок 4.4, які пов’язані з кістками частин тіла автомобіля, для того щоб правильно активувати поведінку підвіски авто, дверних завіс, багажнику, та замків.

Рисунок 4.4 — Список з’єднувачів в Car_BP

Якщо користувачеві не потрібна логіка відкривання дверей та багажника, він можете видалити Car Doors і Trunk Opening логіку в Car_BP, рисунок 4.5.

Потім із компонентів потрібно видалити непотрібні з’єднувачі.

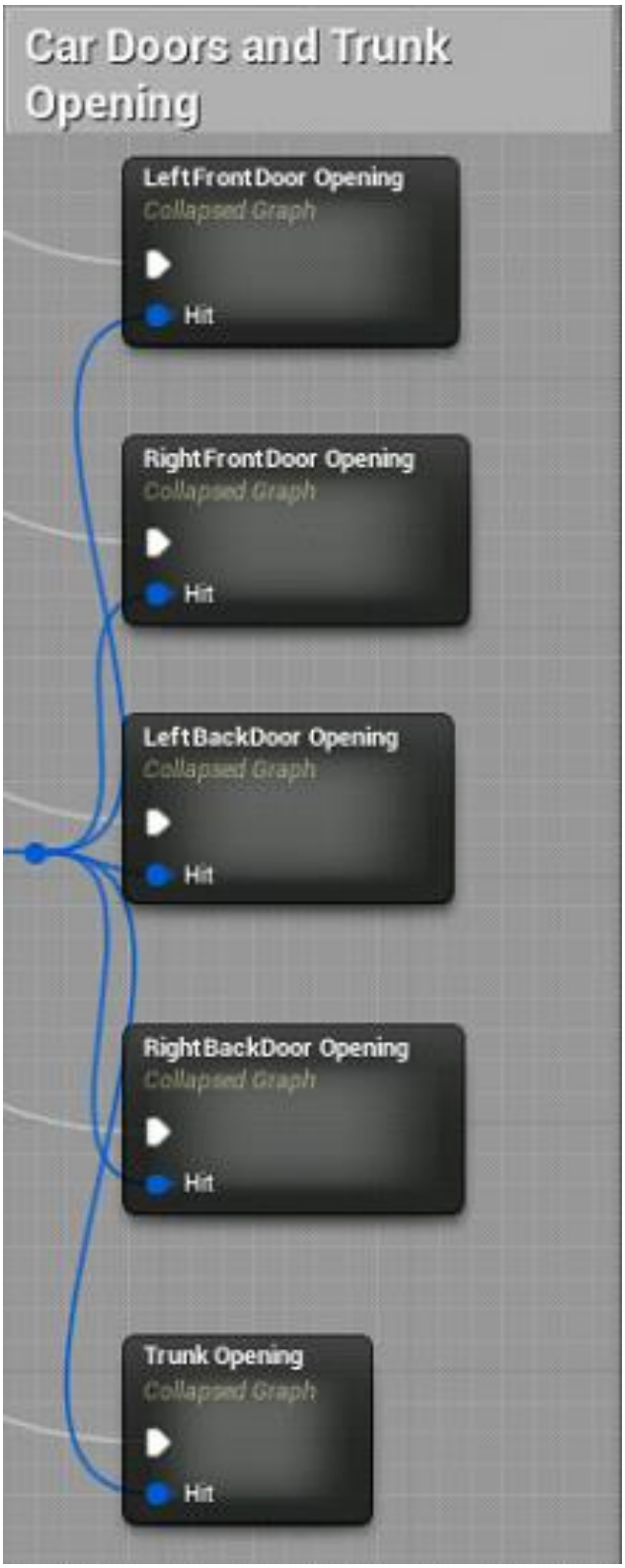


Рисунок 4.5 — Логіка Car Doors і Trunk Opening в Car_BP

Зрештою, потрібно відрегулювати положення/кут обертання всіх мешів, обмежень та інших компонентів.

Можлива зміна поведінки підвіски автомобіля. Наприклад, зробити її більш м'якою або, навпаки, жорсткішою.

Система також дозволяє змінювати характеристики фар автомобіля:

- колір
- потужність
- кут розсіювання світла

4.4 Сценарій роботи системи

На рисунках 4.1 - 4.11 продемонстрована робота системи на прикладі стандартної машини та базової шаблонної рушниці

Рисунок 4.1 — Машина в стані спокою. Без ушкоджень.

Рисунок 4.2 — Машина с простріленим лівим рядом колесом.

На рисунку 4.2 продемонстрована машина с простріленим лівим колесом, як наслідок, вона нахилилась на лівий бік завдяки симуляції підвіски.

Рисунок 4.3 — Машина з імітацією отворів від куль, відкритими дверима від пострілу та потрісканим склом

На рисунку 4.3 продемонстрована машина, в яку був здійснений постріл по лівій двері, як наслідок у цих дверей відстрілений замок, і двері під фізичним впливом відкрились по інерції.

Рисунок 4.4 — Машина з відстреленим лівим дзеркалом заднього виду

Рисунок 4.5 — Машина з пробитим бензобаком та витікаючим пальним

Рисунок 4.6 — Машина з щойно розбитим склом, уламки якого розлітаються навколо

Рисунок 4.7 — Машина з підбитим переднім бампером та пошкодженою фарою

На рисунку 4.7 було здійснено постріл по лівій частині переднього бамперу, через це під силою тяжіння, ліва частина бамперу похилилась вниз.

Рисунок 4.8 — Машина з відкритим від пострілу багажником

Рисунок 4.9 — Машина з щойно відстріленою лівою стоп-фарою, та заднім бампером

На рисунку 4.9 було здійснено постріл по обома частинам заднього бамперу, через це під силою тяжіння, весь бампер звалився на підлогу.

Рисунок 4.10 — Машина з простріленим бензобаком та підпаленим пальним

Рисунок 4.11 — Результат обгоряння кузова автомобіля

На рисунку 4.10-4.11 було здійснено постріл в бензобак, потім по пальному, і показано результати двох етапів горіння - першої секунди і десятої секунди.

ВИСНОВКИ

За результатами виконання дипломної роботи:

1. Розроблено 3Д модель машини, та створено всі необхідні кістки для фізичної взаємодії
2. Створено фізичну модель обстрілу автомобіля, налаштовано відслідковування коллізій та реакцію на них
3. Створено інтерфейс, для взаємодії цієї системи з готовим ігровим проектом
4. Розроблено плагін моделювання обстрілу машини, з можливістю задання фізичного впливу
5. Проведено всі необхідні тестування, та відлагоджено щодо відхилень від дійсних реакцій
6. Створена інструкція користувача, з описаними системними вимогами, доданням нової моделі, інсталлюванням, та системними вимогами

Дана система надає змогу користувачу швидко створити готову ігрову механіку, без великого стороннього втручання, або великих змін. При достатній кваліфікації користувача, він може змінити її для себе, використавши свою модель машини, або скорегувавши систему під себе, використовуючи інструкцію користувача.

В процесі роботи були використані такі програмні продукти як Unreal Engine 4 та Blender3D. Програмний код було написано візуальною мовою програмування Blueprint.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unreal Engine 4 Documentation // https://docs.unrealengine.com/en-US/index.html
2. Udemy Courses // The Complete Cascade Beginners Guide - Learn VFX in Unreal // https://www.udemy.com/course/3dmotive-the-complete-cascade-beginners-guide-learning-vfx-in-udk/
3. David Nixon Beginning Unreal Game Development // Foundation for Simple to Complex Games Using Unreal Engine 4
4. Uengine // Русскоязычное сообщество Unreal Engine 4 // https://uengine.ru/docs
5. Unity User Manual // https://docs.unity3d.com/Manual/index.html

ДОДАТОК А

Моделювання та візуалізація процесу
пошкодження автомобілів

Специфікація

УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б

Аркушів 1

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	Записка.doc	Пояснювальна записка
Компоненти		
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	CabinItem_BP.uasset	Модель інтер’єрних предметів
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	Car_BP.uasset	Модель автомобіля
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	Carlight_BP.uasset	Модель фар автомобіля
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	Mirror_BP.uasset	Модель зеркал автомобіля
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	ShootingInterface.uasset	Інтерфейс стрільби
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	Wheel_BP.uasset	Модель колеса автомобіля
УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б	Windshield_BP.uasset	Модель скла автомобіля

ДОДАТОК Б

Моделювання та візуалізація процесу
пошкодження автомобілів

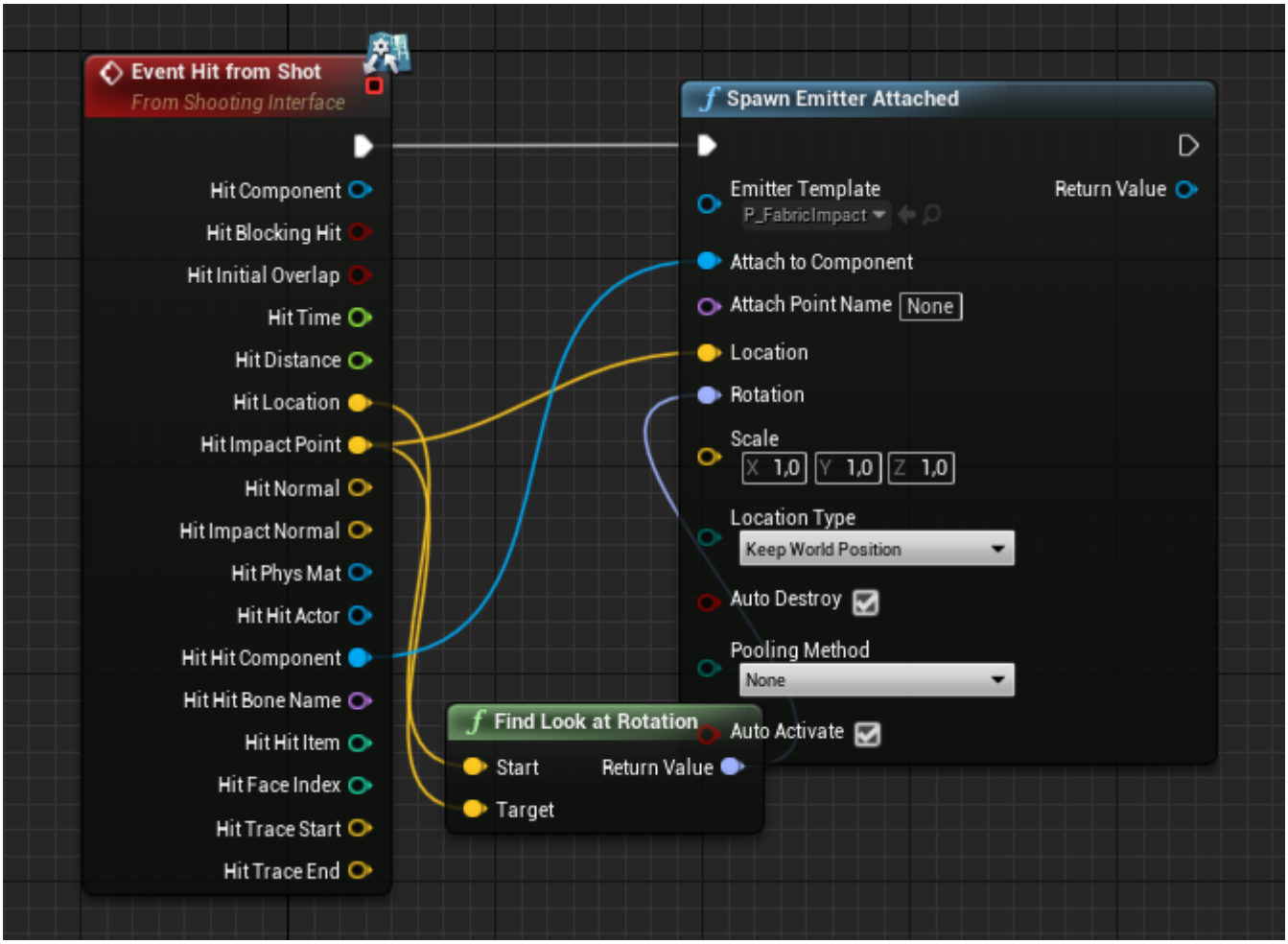
Логіка програми

УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б

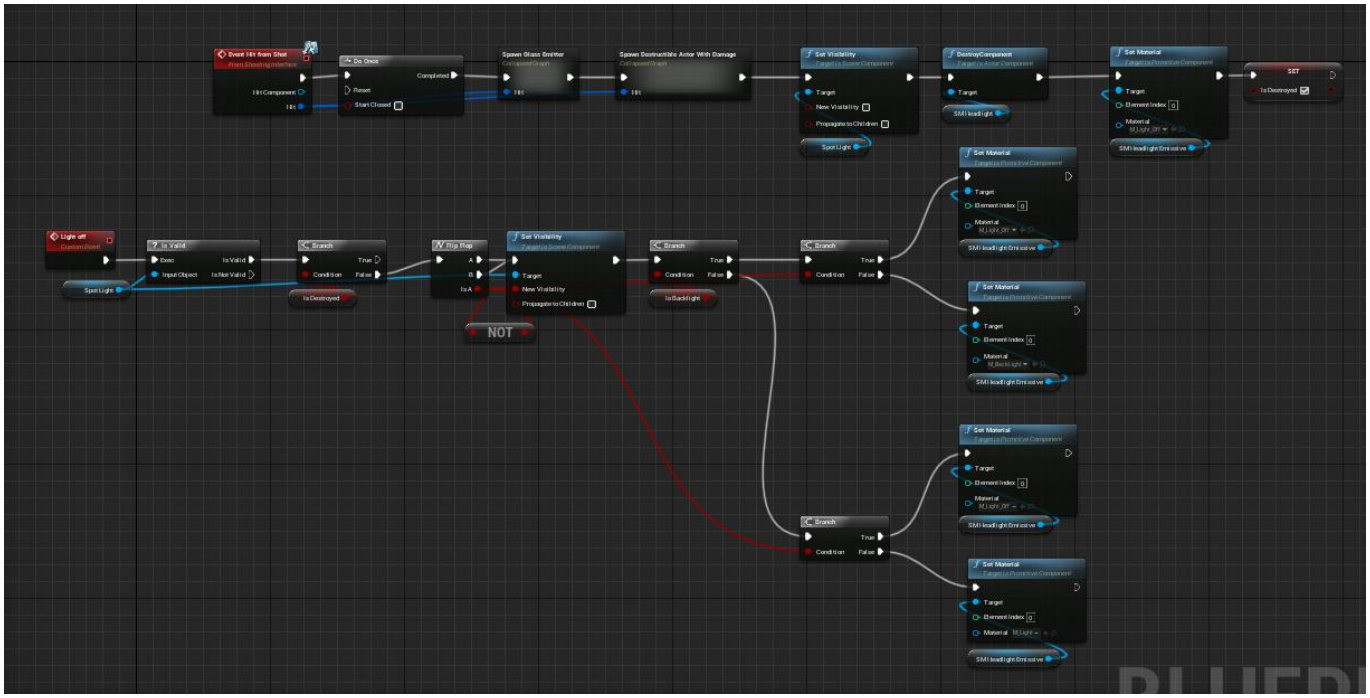
Аркушів 3

Київ 2020

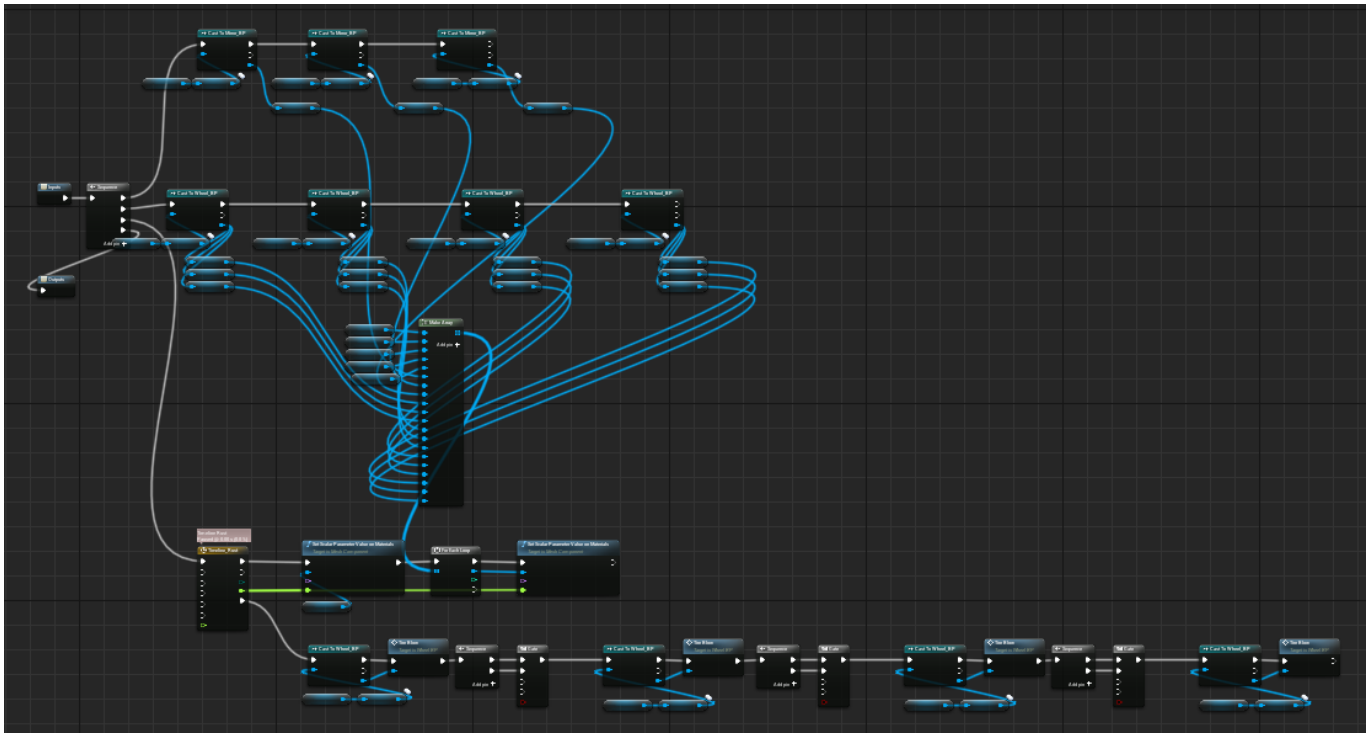
CabinItem_BP



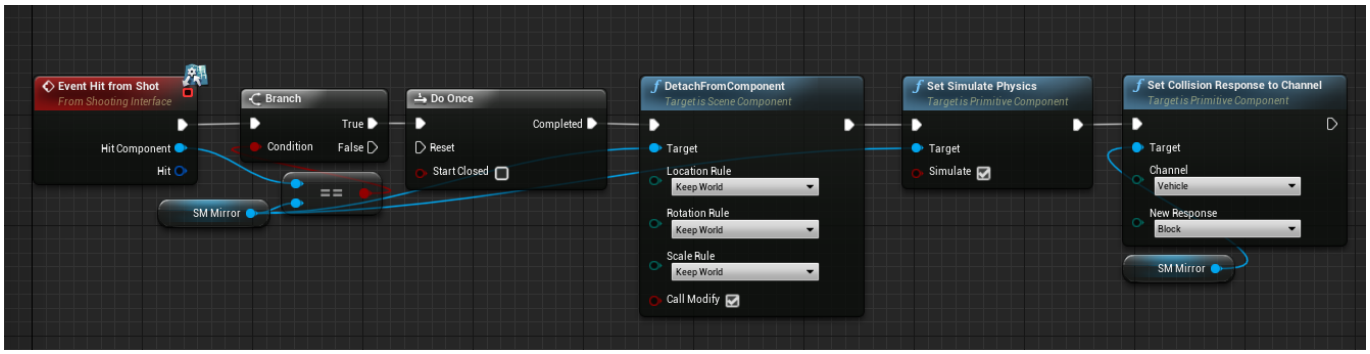
Car_BP



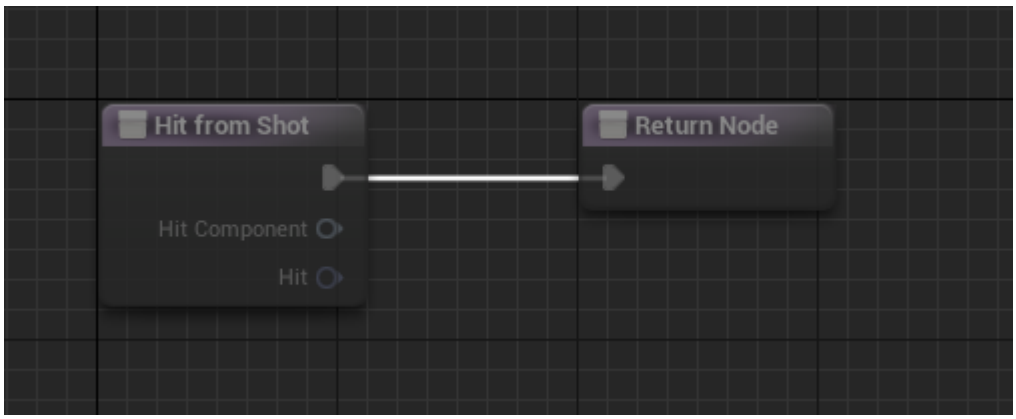
Carlight_BP



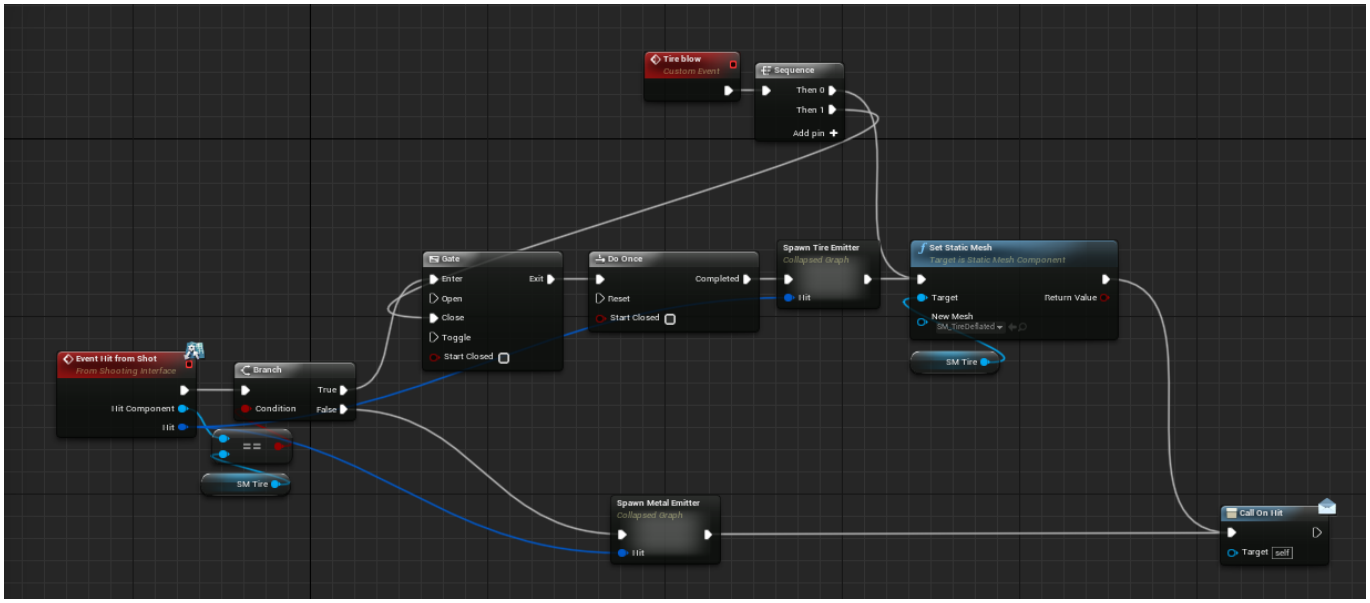
Mirror_BP



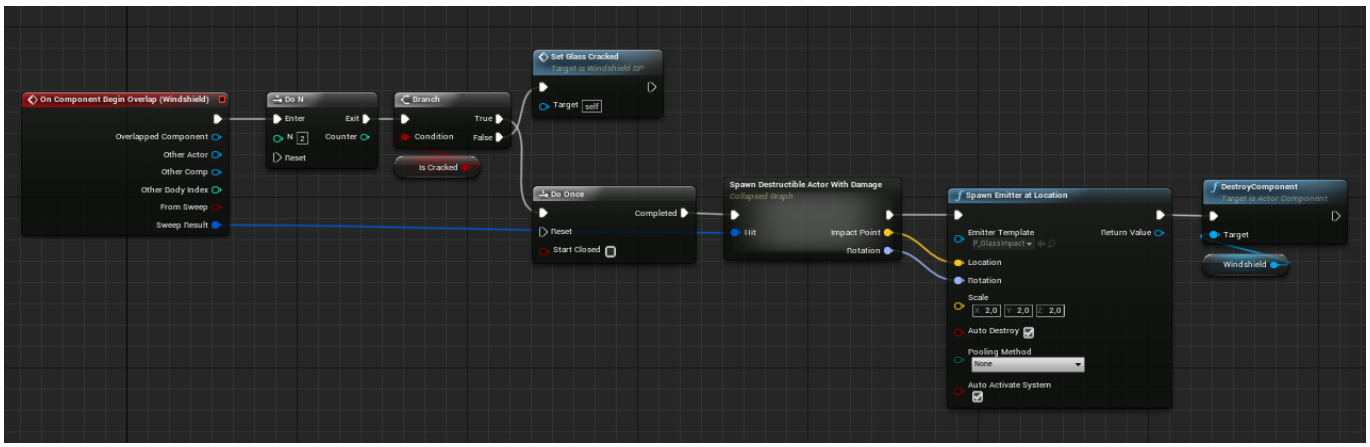
ShootingInterface



Wheel_BP



Windshield_BP



ДОДАТОК В

Моделювання та візуалізація процесу
пошкодження автомобілів

Опис програмного коду

УКР. КПІ ім. Ігоря Сікорського_ТР62130_20Б

Аркушів 2

Київ 2020

В.1 Файл моделі CabinItem_BP.uasset

Файл містить модель поведінки предмету інтер’єру автомобіля, яка містить логіку реакції на постріл, яка створює VFX при пострілі в цей об'єкт. Саму візуальну модель можна змінити в налаштуваннях цього файлу.

В.2 Файл моделі Car_BP.uasset

Файл містить модель автомобіля, яка містить візуальну модель автомобіля, всю основну логіку та логіку реакції на постріли, яка створює VFX при пострілі по кузову, викликає процес запалення автомобіля, а також відкріпляє всі потрібні об’єкти від себе (наприклад дзеркало) і управляє ними.

В.3 Файл моделі Carlight_BP.uasset

Файл містить модель поведінки автомобільних фар, яка містить випромінювачі світла, їх налаштування, логіку реакції на постріл, яка створює VFX при пострілі в цей об'єкт та виключає світло при попадання по ній. Саму візуальну модель можна змінити в налаштуваннях цього файлу. Також в цій моделі передбачені швидкі налаштування для ламп задніх фар, задля коректного інвертування координат.

В.4 Файл моделі Mirror_BP.uasset

Файл містить модель поведінки дзеркала автомобіля, яка містить логіку реакції на постріл, яка відкріпляє дзеркало від автомобіля при пострілі в цей об'єкт. Саму візуальну модель можна змінити в налаштуваннях цього файлу. Також в цій моделі передбачені швидкі налаштування для лівого\правого\центрального дзеркала, задля коректного інвертування координат.

В.5 Файл інтерфейсу ShootingInterface.uasset

Файл містить інтерфейс системи, який використовується для підключення готової системи до ігрового застосунку, та швидкої інтеграції з нею. Інтерфейс не містить логіки, і використовується лише як об’єкт наслідування інших об’єктів.

В.6 Файл моделі Wheel_BP.uasset

Файл містить модель поведінки колеса автомобіля, яка містить логіку реакції на постріл по шині та диску, яка імітує спуск шини, та викликає фізичну реакцію самого автомобіля. Також в цій моделі передбачені швидкі налаштування для лівого там правого колеса, задля коректного інвертування координат.

В.7 Файл моделі Windshield_BP.uasset

Файл містить модель поведінки скла автомобіля, яка містить логіку реакції на постріл, яка створює VFX при пострілі по склу, а потім розбиває його. Саму візуальну модель можна змінити в налаштуваннях цього файлу. Також в цій моделі передбачені швидкі налаштування для лівого\правого\заднього скла, задля коректного інвертування координат.